Modeling Methodology

Nick Singer

10 January 1996

Agenda

Static Modeling of ECS "Push"

Dynamic Modeling

Combined Analytical Queuing Network / Petri Net (CAP) Model

End-to-End Queuing Model

Agenda

Static Modeling of ECS “Push”

- Scope
- Inputs to Static Modeling
- Static Modeling Activities
- Outputs from Static Modeling
- What Happens to the Outputs?

Dynamic Modeling

Combined Analytical Queuing Network / Petri Net (CAP) Model

End-to-End Queuing Model

Scope of Static Modeling

Static modeling is:

- **Applied to first-time “push” processing, i.e. AHWGP PGEs**
- **Used to gain quick insights into the average and busy-day magnitudes of push processing CPU and I/O loads in SDPS**
- **The first step toward SDPS sizing**

Inter-DAAC traffic is handled by a separate static model

Static modeling does *not* (currently) include:

- **V_0 loads**
- **User “pull” loads**
- **Distribution of products**
- **Hardware performance characteristics**
- **Disk sizing**
- **System dynamics**
- **Process/file dependencies**

Inputs to Static Modeling

Technical Baseline

- Operating hours by site (by calendar quarter = “epoch”)

Process Description file

- Comes from AHWGP via Technical Baseline
- Format: Excel spreadsheet
- By (epoch, instrument, PGE), characterizes load on system
  - I/O volumes
  - CPU
  - Frequency of invocation

Static Modeling Activities

Sort Process Description file by epoch, then by instrument

Calculate average MFLOPS for each PGE via $\mathrm{MFLOPS} = \dfrac{\mathrm{MFLO/invocation} \times \mathrm{Invocations/day}}{\mathrm{Number\_of\_operating\_seconds / day}}$

Calculate average I/O bandwidth requirements for each PGE

- Staging
- Destaging, etc.

Accumulate results for each instrument (by site, by epoch)

Perform analysis for "busy day":

- If PGE invocation rate < 1 (per day), then increase it to 1
- Recalculate everything, as above

Outputs from Static Modeling

Outputs are Excel files

Analyses are for average-day and busy-day

Summaries show for each instrument (by epoch, by site):

- **Number of PGE invocations per day**
- **Total MFLOPS required**
- **I/O bandwidth requirements (MB/sec)**
  - **Local to processing**
  - **Host-attached backplane**
  - **Combinations of Staging and Destaging I/O**

Simple multipliers may be applied to account for reprocessing

Excursions may be performed to consider the effects of different operating hours at a site

What Happens to the Outputs?

Used by the Performance Modeling Team to get rough estimates of processing loads by instrument

- **If at odds with initial perceptions, this is an error-correcting opportunity.**

Used by Multi-Release Support personnel to begin processor and LAN sizing analyses

- **Average- and busy-day loads give an initial estimate of how many processors will be needed to meet timeliness performance requirements. These may be used as constraint inputs to the dynamic model.**
- **May be published as-is (DID 305, Appendix E, Table E-1)**

Used by DAAC Planners/Schedulers

- **How many PGE invocations per day are there? Is this a feasible load to place on automated and/or manual planners/schedulers?**

Agenda

Static Modeling of ECS “Push”

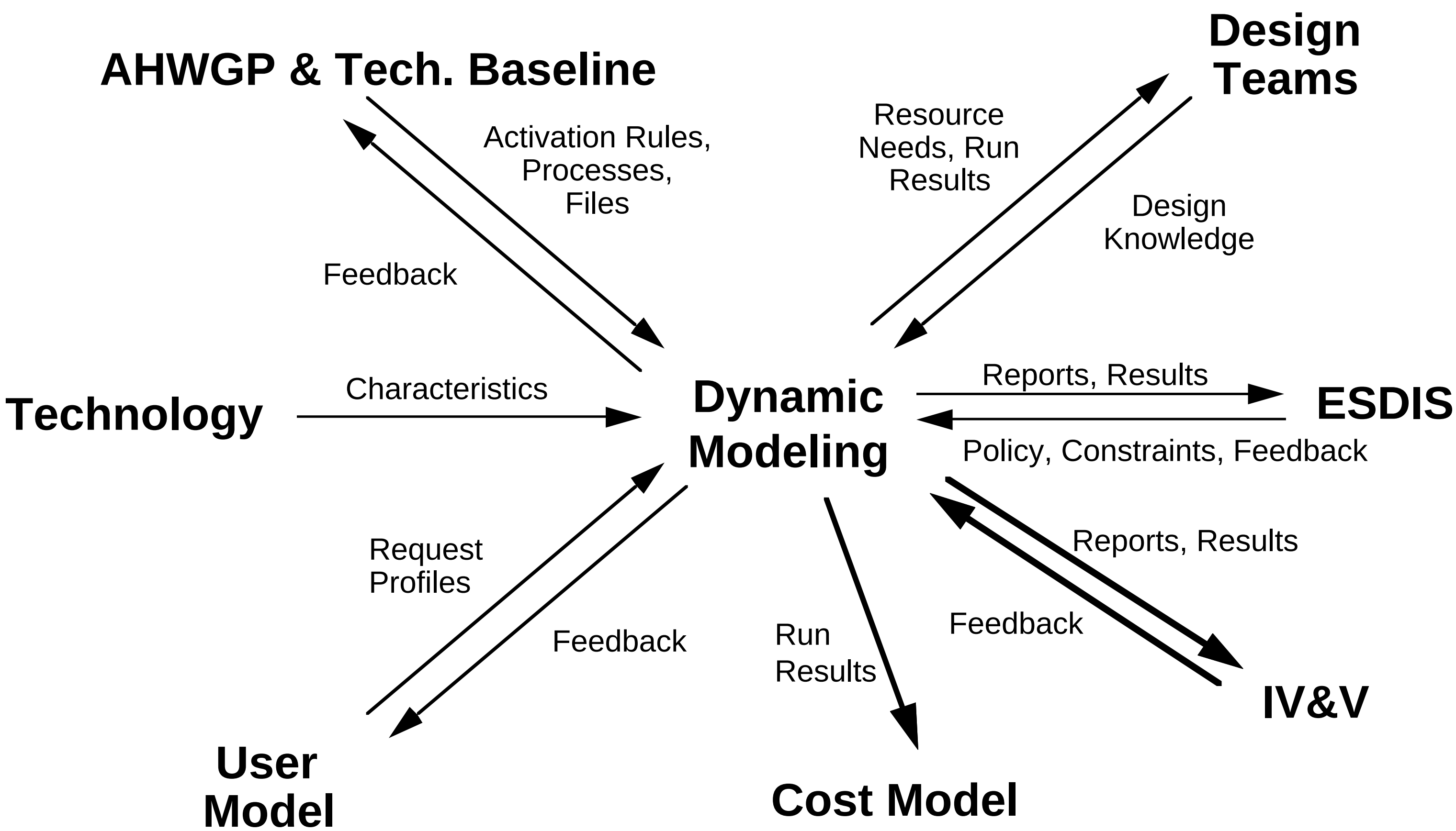
Dynamic Modeling

- **Interaction of Dynamic Modeling with Other Entities**
- **Model Context Diagram**
- **Dynamic Model’s Implementation of a DAAC**
- **Major Inflows Modeled**
- **Dynamic Model does not Model all Possible Resources/Constraints**
- **Modeling Activities**
- **Outputs from Dynamic Modeling**
- **What Happens to the Outputs?**

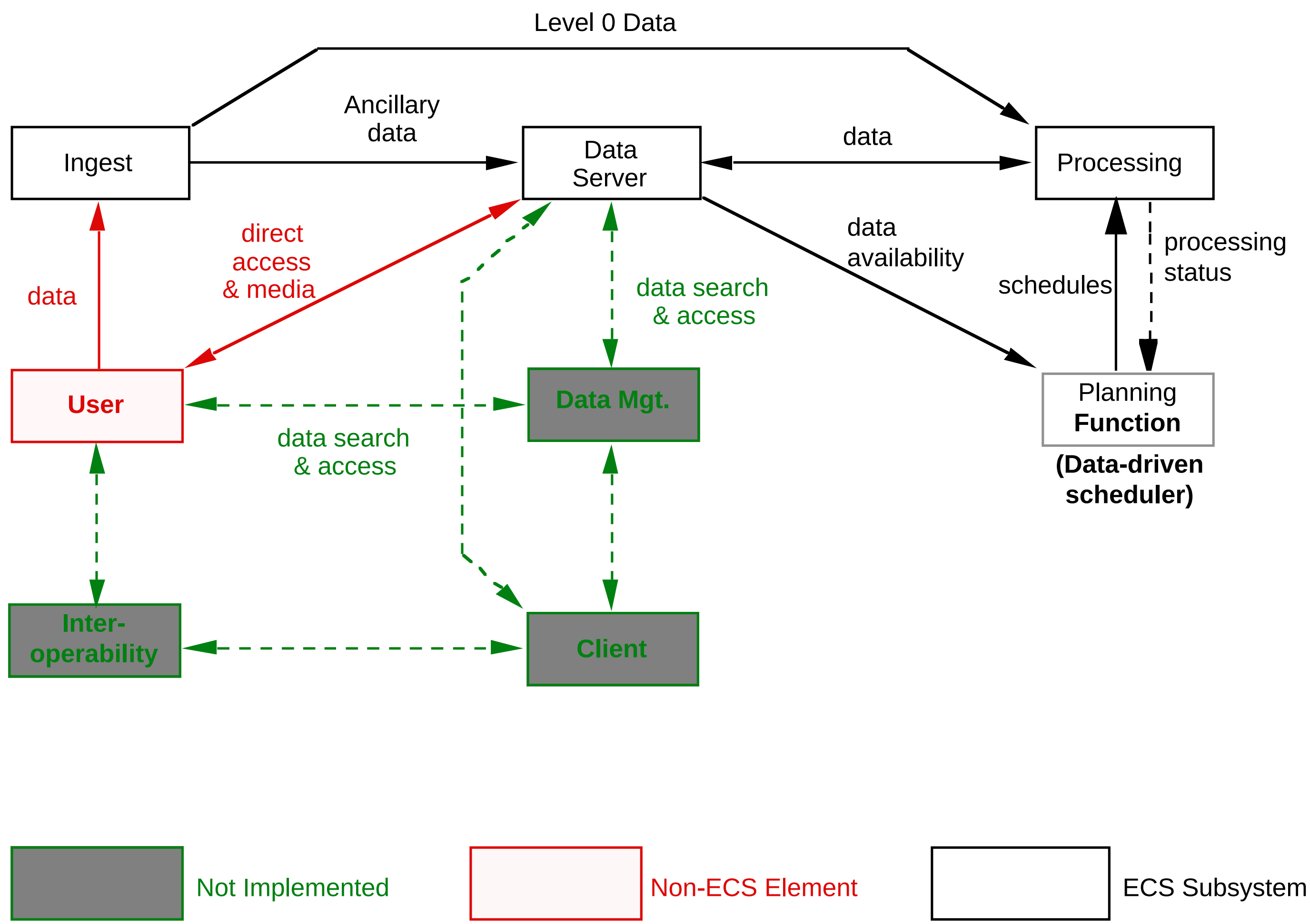
Combined Analytical Queuing Network / Petri Net (CAP) Model

End-to-End Queuing Model

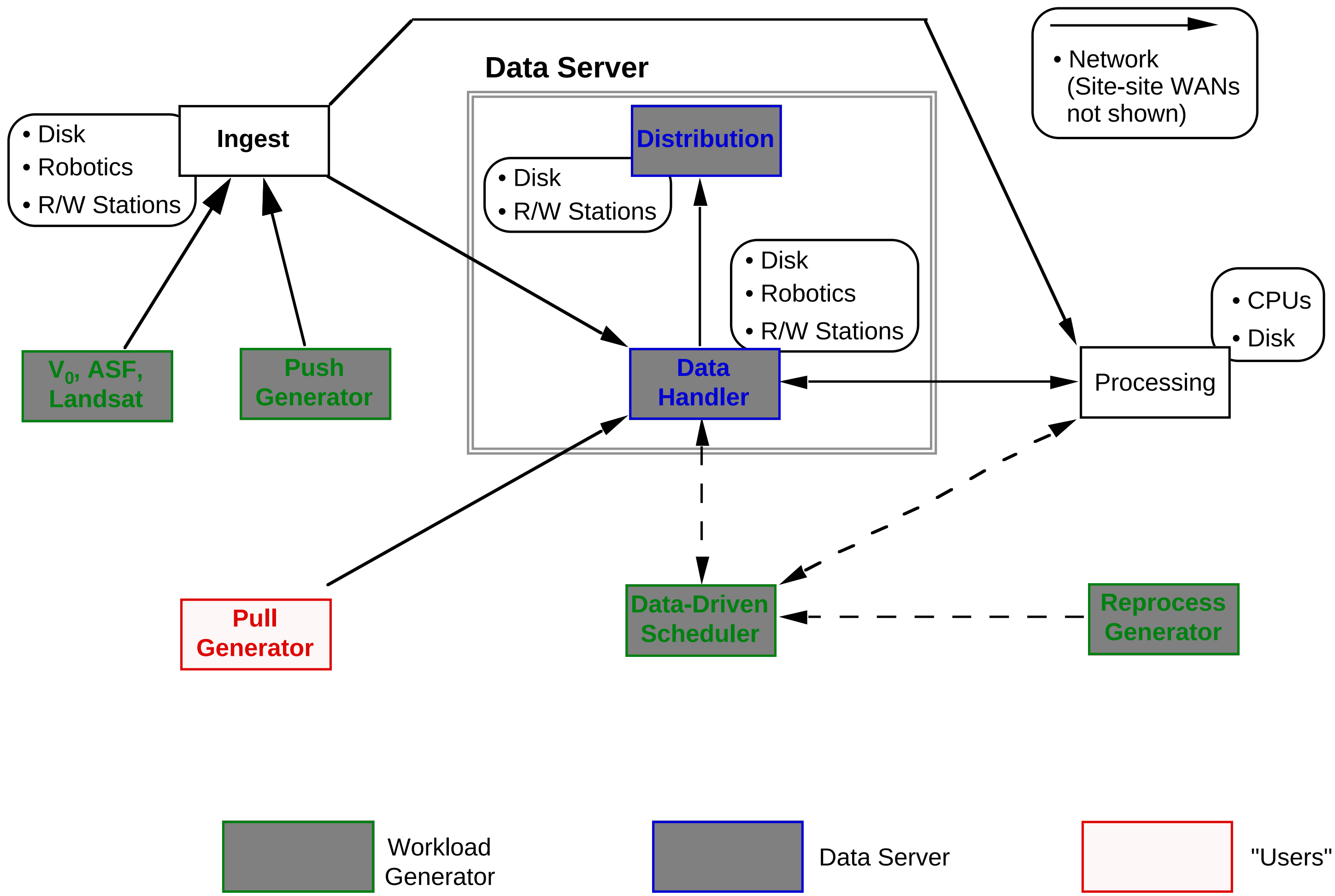
Interaction of Dynamic Modeling with Other Entities



Model Context Diagram (Overall)



Dynamic Model's Implementation of a DAAC



Major Inflows Modeled

Ingest

- L_0 data from EDOS and SDPF
- Landsat data transfer (EDC only)
- Ancillary data from SCFs, ADCs, ODCs, and users
- Reprocessing (optional)

Data Server

- V_0 Ingest
- Radar data transfer (ASF only)
- TSDIS data transfer (*was* VIRS at GSFC and PR, TMI & GV at MSFC; new baseline will reflect changes at MSFC)

Multiple Subsystems

- User requests

Dynamic Model does not Model all Possible Resources/Constraints

Resources currently not tracked:

- Subsystems other than Ingest, Data Server, Processing, and Distribution
- Memory allocated within a processor
- I/O channels*
- Disk controllers*
- Processor overhead
  - job initialization & termination, swapping, virtual memory operations, ...

It is not reasonable to simulate

- items whose operations or inputs are not known or are known imprecisely,
- in so much detail that model execution time approaches that of the real system.

* Currently adding these resources to the dynamic model

Modeling Activities

Use:

- AHWGP inputs to produce time-phased, interdependent “push” demands
- User Model inputs and assumptions to produce scaled “pull” demands
- Tech Baseline for V_0 processing; reprocessing assumptions; each site’s operating hours
- ECS Designs to characterize architectural interconnections
- Technology assumptions (benchmarking and vendor-supplied) to set component performance characteristics

Set assumed constraints on resources

Use discrete-event simulation model (BONeS) to compute dynamic system response (see next slide)

Examine model outputs for unexpected results; analyze to find cause; correct and rerun if necessary

Outputs from Dynamic Modeling

Outputs are plots and tables; may be turned into Excel files

For each resource (pool), model can show:

- Average and peak utilization
- Timeline plots (every 15 minutes) of
  - Utilization
  - Queue length
  - Response time

For each transaction type, model can show timeline plots of system response time

Probes can be inserted to measure and report just about anything desired

What Happens to the Outputs?

Performance Modeling Team **gets fairly precise estimates of processing loads by instrument**

- **Provide feedback to AHWGP, instrument teams**
- **If at odds with initial perceptions, this is an error-correcting opportunity**
- **Model results may influence instrument teams to redesign their algorithms**

Hardware Designers **refine processor, data server, and network sizing analyses**

- **Runs are made using realistic constraints on**
  - **Numbers of processors for each instrument**
  - **Network bandwidths**
  - **Robots**
  - **Read/write stations, etc.**
- **If performance requirements are met, then sizing is at least adequate.**

Agenda

Static Modeling of ECS "Push"

Dynamic Modeling

***C**ombined **A**nalytical Queuing Network / **P**etri Net (CAP) Model*

- **Background, Purpose, Scope**
- **Technical Rationale**
- **Inputs to CAP**
- **CAP Modeling Activities, Outputs**
- **Two-Level Model Overview**

End-to-End Queuing Model

Background, Purpose, Scope

GMU, COLA, U. of Delaware, and U. of New Hampshire performed a scientific and technical evaluation ("independent architecture study") of ECS, dated August 31, 1994: "The GMU ECS Federated Client-Server Architecture." Professor Daniel Menascé was the lead for Part 3, Chapter 5, Performance Modeling.

HITC contracted with Prof. Menascé to design an appropriate *analytical* model of ECS performance and produce a Borland Pascal implementation.

Model has two major applications:

- Run in parallel with dynamic model ("what-if" excursions, cross-validation)
- For end-to-end modeling

Technical Rationale

Discrete-event simulation models can provide great flexibility and fidelity of workload characterization and system response, as they evolve over time. The cost for this is:

- **Long model development time**
- **Difficult model verification (let alone validation)**
- **Long run times (hours to days)**
- **Output (may be) difficult to interpret**

Queuing network models capture almost the same level of detail of workload and processing. Such models:

- **Produce *steady-state* answers very quickly (in seconds)**
- **Are simple to construct and debug**
- **Do not capture system transients**
- **Do not capture intricate interdependencies**

Petri nets:

- **Can capture the data-driven process activations of ECS SDPS**
- **Our Petri net is a Deterministic Timed Petri Net (DTPN) with Nonatomic Firing**

Inputs to CAP

Uses essentially the same input files as the Dynamic Model

- **File and process description files**
- **Process input and output lists**
- **Pull workload description**
- **Subsystem resource characterizations**
  - **Processing**
  - **Data Handler**
  - **Ingest**
  - **Distribution**
- **Networks**
  - **Resources between subsystems**
  - **Bandwidth between subsystems**
  - **Site-to-site bandwidths**

Models same resources as Dynamic Model, *plus* queuing for disk controllers

CAP Modeling Activities, Outputs

Capture process-file dependencies in a DTPN. The QN model will "tell" it the queuing times to add to the execution times.

- Process execution time is assumed to be deterministic
- Queuing time at the various system resources is assumed to be reflected in the process execution time
- Queuing/execution times come from the queuing network model
- First time through, zero contention is assumed

Capture the queuing and resource contention aspects in a QN model. The DTPN model will "tell" it the arrival rates at each resource.

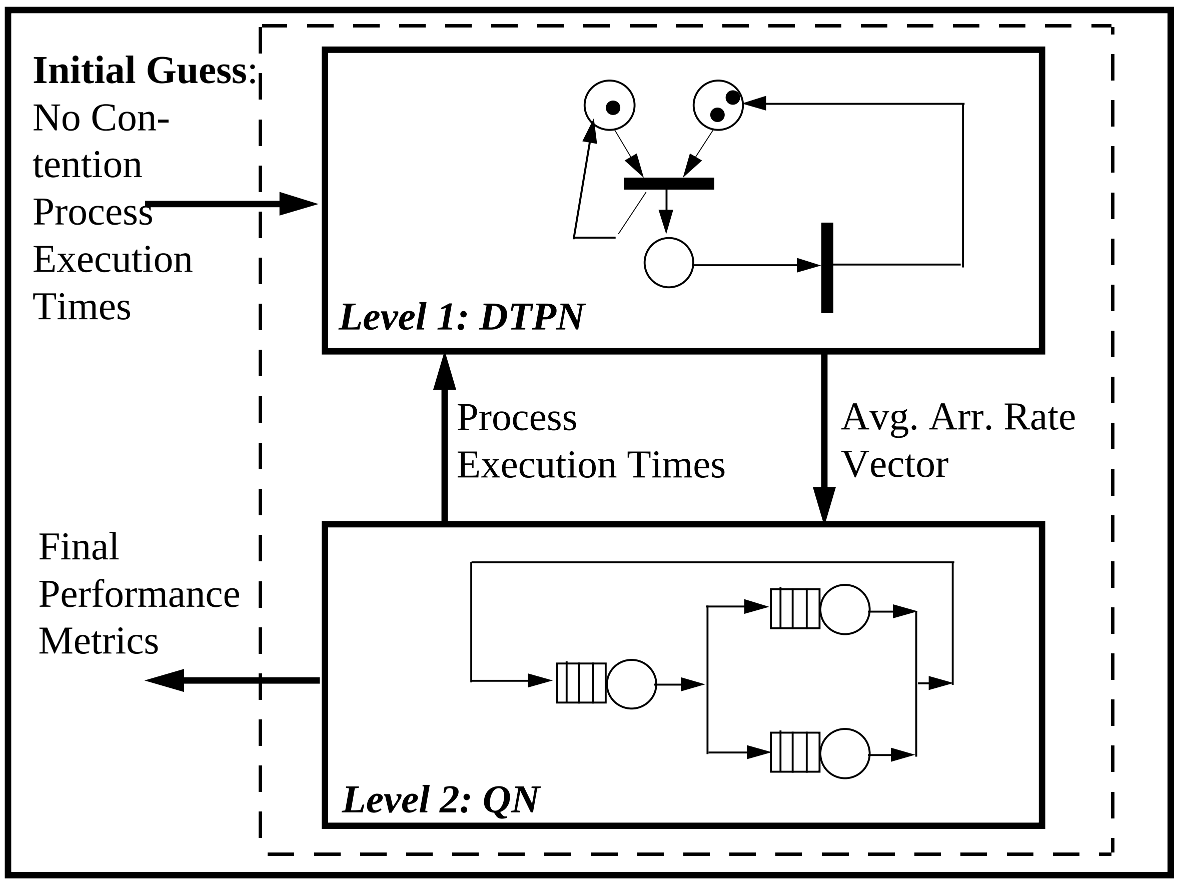
- For each process type, arrival rate of process at resource = Number of processes at resource / Process execution time at resource [Little's Law]

Iterate between the two models until queuing times converge

Use QN model one last time to compute final performance metrics:

- Process execution times (in queue and on resource)
- Throughputs
- Resource utilizations
- Queue lengths

Two-Level Model Overview



Agenda

Static Modeling of ECS "Push"

Dynamic Modeling

Combined Analytical Queuing Network / Petri Net (CAP) Model

End-to-End Queuing Model

- **Scope of End-to-End Modeling**
- **Inputs to End-to-End Modeling**
- **Modeling Activities**
- **Outputs from End-to-End Modeling**
- **What Happens to the Outputs?**

Scope of End-to-End Modeling

Applied to (nearly) all processing:

- **Push**
- **Pull**
- **V_0 loads**
- **Distribution of products**
- **Infrastructure loads**

Inputs to End-to-End Modeling

Threads

- Composed of thread elements
  - Software executable
  - Other resource call
- Partitioning of possible work flows through ECS
- Should account for nearly all work done by ECS, in each subsystem
- Each thread and/or each thread element should have a multiplier corresponding to frequency of invocation
- For each SW executable (CSCI or CSC):
  - Nominal MI or MFLO per execution
  - Which HWCI(s) it runs on
  - What executables this executable calls
  - What executables call this executable
  - Prob. that this executable needs to be loaded from disk (vs. already present in RAM)
  - which disk HWCI(s) it resides on
- For other resource calls
  - MB moved over network
  - Robots, read/write stations, and tape drives: Tape mounts, files & MB read/written
  - Disk accesses, files & MB read/written

Inputs to End-to-End Modeling (cont'd)

Subsystem characterizations

- **For network HWCI:**
  - Protocol
  - Bandwidth (Nominal MB/sec)
  - Connecting which other HWCIs
- **For processing HWCI**
  - Number of (independent) processors
  - Nom. MIPS/MFLOPS (for each processor)
  - I/O bandwidth
  - Attached to which: disks, networks
- **For "tape" HWCI:**
  - Capacity (GB) per "reel/cartridge"
  - Nominal transfer rate (MB/sec)
  - # ports (assume each has an independent controller)
  - Nominal spin time to get to dataset
  - Nominal rewind time

Inputs to End-to-End Modeling (cont'd)

- For disk HWCI:
  - Total capacity (GB)
  - Nominal transfer rate (MB/sec)
  - # ports (assume each has an independent controller)
  - Nominal latency time
- For robotics HWCI:
  - Nominal time to grab/replace, travel, mount/demount
  - Number of robots (or arms, if approp.)
  - Attached to which "tape" HWCIs
- HWCIs are indexed by :
  - Location (i.e. DAAC)
  - Subsystem (e.g. DSS, MSS, etc.)
  - Index (e.g. cluster number, or whatever differentiation makes sense)

Modeling Activities

Read input files

Build model/Generate devices

Compute and report service demands on devices

- Here is where we introduce results from:
  - Finer-grain models, e.g. Dynamic Model
  - Transaction estimating
  - Benchmarking

Solve queuing network

Generate outputs

- Which may also be fine-tuned from benchmarking

Outputs from End-to-End Modeling

Outputs are text files, ready to be imported into Excel

Input Echo

- **Baseline parameters**
- **Component characteristics**

Busy Processors

- **Average number of busy processors per site/subsystem/cluster**

Busy Read/write stations

- **Average number of busy read/write stations per site/subsystem/cluster**

Disk Utilization

- **Percentage disk utilization per site/subsystem/cluster**

Network Utilization

- **LAN utilization by site**

Outputs from End-to-End Modeling (cont'd)

End-to-end thread execution times (average)

Thread time profile (where does the thread spend its time)

Thread throughputs (activations/day)

Pull workload response time vs. arrival rate

What Happens to the Outputs?

Used to gain insights into the total loads on each subsystem, and to evaluate the expected response of the system to a given load

- **Utilizations**
- **Throughputs**
- **Response Times**

Forms a basis of estimate (BOE) for subsystem sizing decisions

Designers compare predicted system response times with stated requirements and policy and redesign accordingly

Used as part of the design validation process